
MONGODB VALIDATION

Phó Nghĩa Văn



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



MYC  **DER**

timviec  **it.com**

MỤC TIÊU BÀI HỌC

- Thực hiện validation với MongoDB
- **Ghi chú:** sử dụng lại source code của buổi học về MongoDB trước đó
- TLTK: <https://mongoosejs.com/docs/validation.html>



REQUIRED

- Sử dụng **required** để bắt buộc người dùng phải nhập một filed nào đó, nếu ko nhập sẽ báo error

```
const mongoose = require('mongoose');

const CourseSchema = new mongoose.Schema({
  _id: String,
  name: {type: String, required: true},
  author: String,
  tags: [String],
  date: {type: Date, default: Date.now},
  isPublished: Boolean,
  price: {
    type: Number,
    required: function(){
      return this.isPublished
    }
  }
});
```

Không sử dụng
arrow function,
vì arrow function
không có **this**

```
const Course = mongoose.model('Course', CourseSchema);
module.exports = Course;
```

REQUIRED

Kiểm tra kết quả:

- Sử dụng **async/await** và **try-catch** thay cho promise (hiện ít thông báo lỗi hơn)
- course thêm mới vào **không có** field name
- Chạy lệnh **nodemon server.js**

```
async function createCourse(){  
  const newCourse = new Course({  
    _id: "abc123123",  
    author: "Hackagon",  
    tags: ["NodeJS", "ExpressJS"],  
    isPublished: true,  
  });  
  
  try { không có field name và price  
    const result = await newCourse.save()  
    console.log(result)  
  } catch (error) {  
    console.log(error.message)  
  }  
}  
  
createCourse();
```



```
Course validation failed: price: Path `price` is required.,  
name: Path `name` is required.  
Connected to MongoDB
```

com

MINLENGTH & MAXLENGTH

- Sử dụng validation với **minlength** và **maxlength**

Course validation failed: name: Path `name` (`js`) is shorter than the minimum allowed length (5).

```
const CourseSchema = new mongoose.Schema({
  _id: String,
  name: {
    type: String,
    required: true,
    minlength: 5,
    maxlength: 255
  },
  author: String,
  tags: [String],
  date: {type: Date, default: Date.now},
  isPublished: Boolean,
  price: {
    type: Number,
    required: function(){
      return this.isPublished
    }
  }
})
```

ENUM

- Dùng **enum** khi muốn người dùng phải khai báo field với các giá trị xác định trước

Course validation failed: category: `network` is not a valid enum value for path `category`.



```
const CourseSchema = new mongoose.Schema({
  _id: String,
  name: {
    type: String,
    required: true,
    minlength: 5,
    maxlength: 255
  },
  category: {
    type: String,
    required: true,
    enum: ['web', 'mobile', 'hacking']
  },
  author: String,
  tags: [String],
  date: {type: Date, default: Date.now},
  isPublished: Boolean,
  price: {
    type: Number,
    required: function(){
      return this.isPublished
    }
  }
})
```

MIN & MAX

- Type **Number** sẽ có validation

Course validation failed: category: `network` is not a valid enum value for path `category`.
price: Path `price` (300) is more than maximum allowed value (200).



MYO

```
const CourseSchema = new mongoose.Schema({
  _id: String,
  name: {
    type: String,
    required: true,
    minlength: 5,
    maxlength: 255
  },
  category: {
    type: String,
    required: true,
    enum: ['web', 'mobile', 'hacking']
  },
  author: String,
  tags: [String],
  date: {type: Date, default: Date.now},
  isPublished: Boolean,
  price: {
    type: Number,
    required: function(){
      return this.isPublished
    },
    min: 10,
    max: 200
  }
})
```


CUSTOM VALIDATOR

- Yêu cầu: mỗi course phải có ít nhất một tag
- Có thể tự tạo validator như sau:

```
tags: [{
  type: [String],
  validate: {
    validator: function(tags){
      return tags && tags.length > 0
    },
    message: "A course must have at least one tag"
  }
}],
```

```
Course validation failed: tags:
A course must have at least one
tag
```


VALIDATION ERRORS

- Có thể thực hiện việc in ra các errors bằng cách duyệt vòng lặp như hình bên

```
Path `name` is required.  
A course must have at least one tag  
Path `price` (300) is more than maximum  
allowed value (200).
```



```
async function createCourse(){  
  const newCourse = new Course({  
    _id: "abc123123",  
    name: "",  
    author: "Hackagon",  
    tags: null,  
    category: "web",  
    isPublished: true,  
    price: 300  
  })  
  
  const result = await newCourse.save()  
  console.log(result)  
}  
catch (error) {  
  for(let index in error.errors){  
    console.log(error.errors[index].message)  
  }  
}
```